

Как написать свой индекс в Tarantool

Бабин Олег

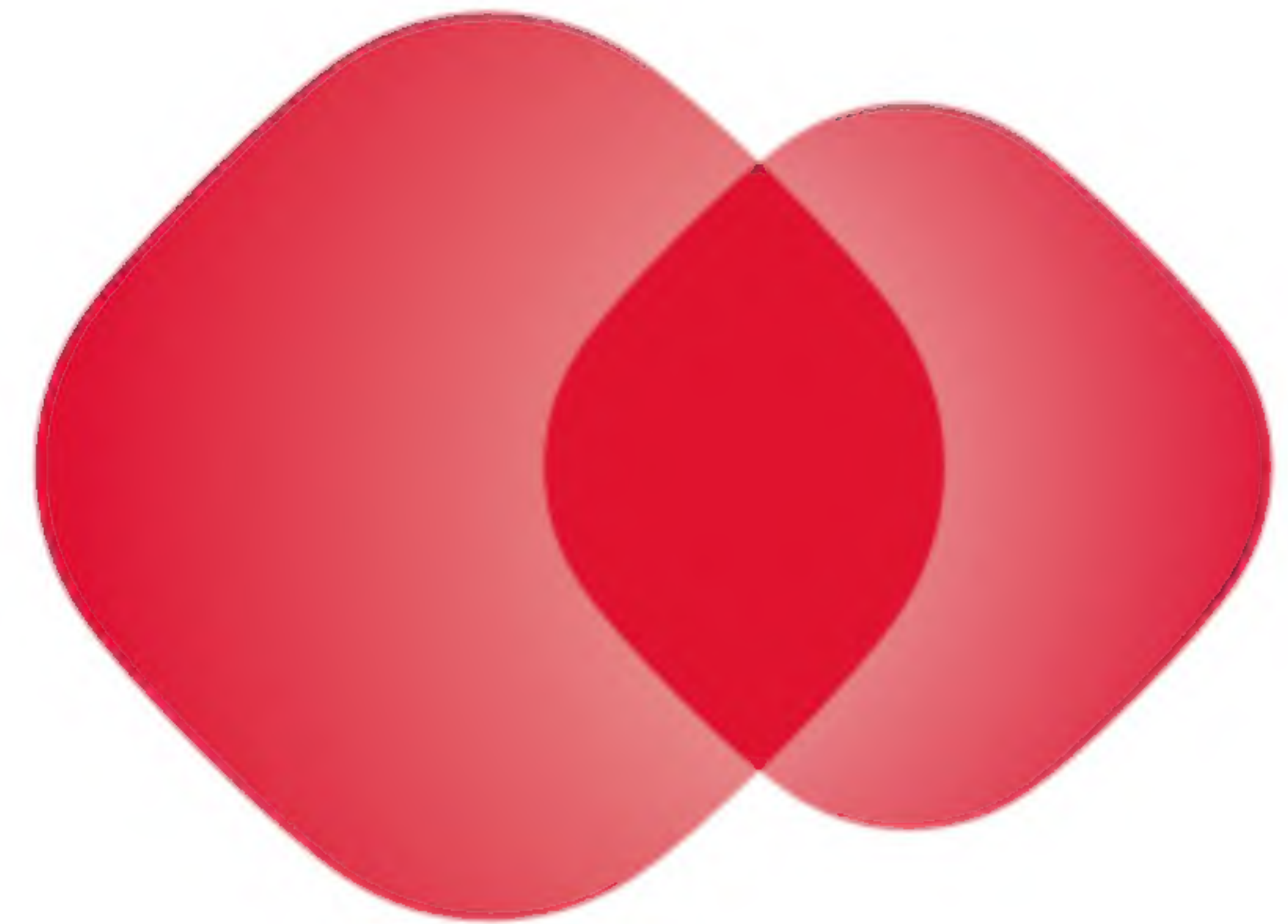


HighLoad++
Весна 2021




Tarantool

- Сервер приложений
- База данных
 - memtx — хранение в памяти
 - vinyl — хранение на диске




Первый запуск




```
1 box.cfg{}
2 box.schema.space.create( 'my_space' )
3 box.space.my_space:create_index( 'primary',
4     {parts = {{field = 1, type = 'unsigned'}}})
5 box.space.my_space:replace({1, 'hello'})
6 box.space.my_space:replace({2, 'world'})
```


Первый запуск



```
1 box.cfg{}
2 box.schema.space.create('my_space')
3 box.space.my_space:create_index('primary',
4     {parts = {{field = 1, type = 'unsigned'}}})
5 box.space.my_space:replace({1, 'hello'})
6 box.space.my_space:replace({2, 'world'})
```


Первый запуск




```
1 box.cfg{}
2 box.schema.space.create('my_space')
3 box.space.my_space:create_index('primary',
4     {parts = {{field = 1, type = 'unsigned'}}})
5 box.space.my_space:replace({1, 'hello'})
6 box.space.my_space:replace({2, 'world'})
```


Первый запуск



```
1 box.cfg{}
2 box.schema.space.create( 'my_space' )
3 box.space.my_space:create_index( 'primary',
4     {parts = {{field = 1, type = 'unsigned'}}})
5 box.space.my_space:replace({1, 'hello'})
6 box.space.my_space:replace({2, 'world'})
```


Первый запуск



```
1 box.cfg{}
2 box.schema.space.create('my_space')
3 box.space.my_space:create_index('primary',
4     {parts = {{field = 1, type = 'unsigned'}}})
5 box.space.my_space:replace({1, 'hello'})
6 box.space.my_space:replace({2, 'world'})
```


Выборка данных



```
1 tarantool> box.space.my_space:select()  
2 ---  
3 - - [1, 'hello']  
4   - [2, 'world']  
5 ...  
6 tarantool> box.space.my_space:select({1},  
7                                     > {iterator = box.index.GT})  
8 ---  
9 - - [2, 'world']  
10 ...
```


Выборка данных



```
1 tarantool> box.space.my_space:select()  
2 ---  
3 - - [1, 'hello']  
4   - [2, 'world']  
5 ...  
6 tarantool> box.space.my_space:select({1},  
7                                     > {iterator = box.index.GT})  
8 ---  
9 - - [2, 'world']  
10 ...
```


Выборка данных



```
1 tarantool> box.space.my_space:select()  
2 ---  
3 - - [1, 'hello']  
4   - [2, 'world']  
5 ...  
6 tarantool> box.space.my_space:select({1},  
7                                     > {iterator = box.index.GT})  
8 ---  
9 - - [2, 'world']  
10 ...
```


Вторичные индексы

```
1 space = box.space.my_space
2 space:create_index('secondary',
3     {parts = {{field = 2, type = 'string'}}})
4 tarantool> space.index.secondary:select({'world'})
5 ---
6 - - [2, 'world']
7 ...
```


Вторичные индексы

```
1 space = box.space.my_space
2 space:create_index('secondary',
3     {parts = {{field = 2, type = 'string'}}})
4 tarantool> space.index.secondary:select({'world'})
5 ---
6 - - [2, 'world']
7 ...
```


Вторичные индексы

```
1 space = box.space.my_space
2 space:create_index('secondary',
3     {parts = {{field = 2, type = 'string'}}})
4 tarantool> space.index.secondary:select({'world'})
5 ---
6 - - [2, 'world']
7 ...
```


Многомерные данные





1 из 6 фото

Смартфон Xiaomi Redmi 9 4/64GB (NFC)

160 отзывов

👍 долгое время работы, объем памяти

Android 10
поддержка двух SIM-карт
экран 6.53", разрешение 2340x1080
4 камеры: широкоугольная (13 МП), сверхширокоугольная, макро, датчик глубины
процессор MediaTek Helio G80
6 606 человек купили этот товар

11 599 Р

от 437 Р/мес

75 предложений от 11 528 Р

Четкий экран, яркости мне хватает
Наклеенная защитная пленка "с завода"....
Андрей В.

Производитель

Apple
BQ
HONOR
HUAWEI
Nokia
OnePlus
OPPO
Philips
realme
Samsung
Xiaomi
ZTE

[Показать все](#)

✓ В продаже

Тип

кнопочный телефон
смартфон

Платформа

Android
iOS

Игровой

Диагональ экрана (точно), "

от 0.66 до 7.6

Диагональ экрана

до 3.4"
3.5"-4.9"
5.0"-5.4"
5.5"-5.9"
6.0"-6.4"
6.5" и больше



1 из 6 фото

Смартфон Apple iPhone 11 128GB

4.7 319 отзывов

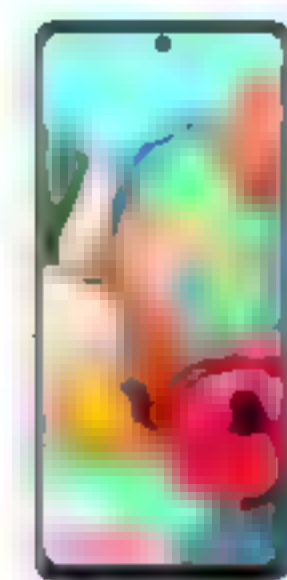
👍 качество фотографий, мощный процессор

iOS 13
поддержка двух SIM-карт (nano SIM+eSIM)
экран 6.1", разрешение 1792x828
двойная камера: 12 МП, 12 МП
процессор Apple A13 Bionic
8 922 человека купили этот товар

55 390 Р

327 предложений от 55 390 Р

Уникальный аппарат, который уверенно
смотрит в будущее, да и сам - уже в...
Ольга К.



1 из 6 фото

Смартфон Samsung Galaxy A71 6/128GB

273 отзыва

👍 экран, объем памяти

Android 10
поддержка двух SIM-карт
экран 6.7", разрешение 2400x1080
4 камеры: 64 МП, 12 МП, 5 МП, 5 МП
процессор Qualcomm Snapdragon 730
6 011 человек купили этот товар

22 650 Р

140 предложений от 22 450 Р

Большой, красочный экран
Время автономной работы Быстрая зарядка...
Максим Т.



Смартфон Xiaomi Redmi 9A

80 отзывов

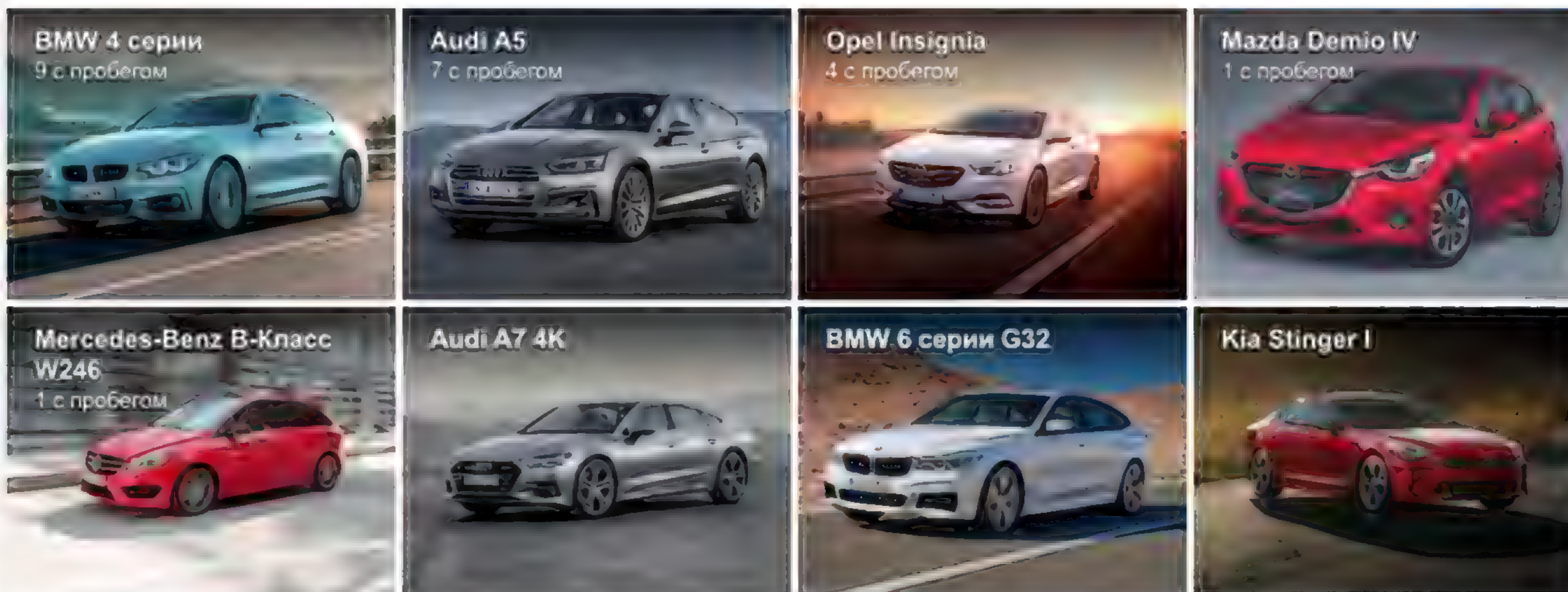
7 450 Р

92 предложения от 7 339 Р

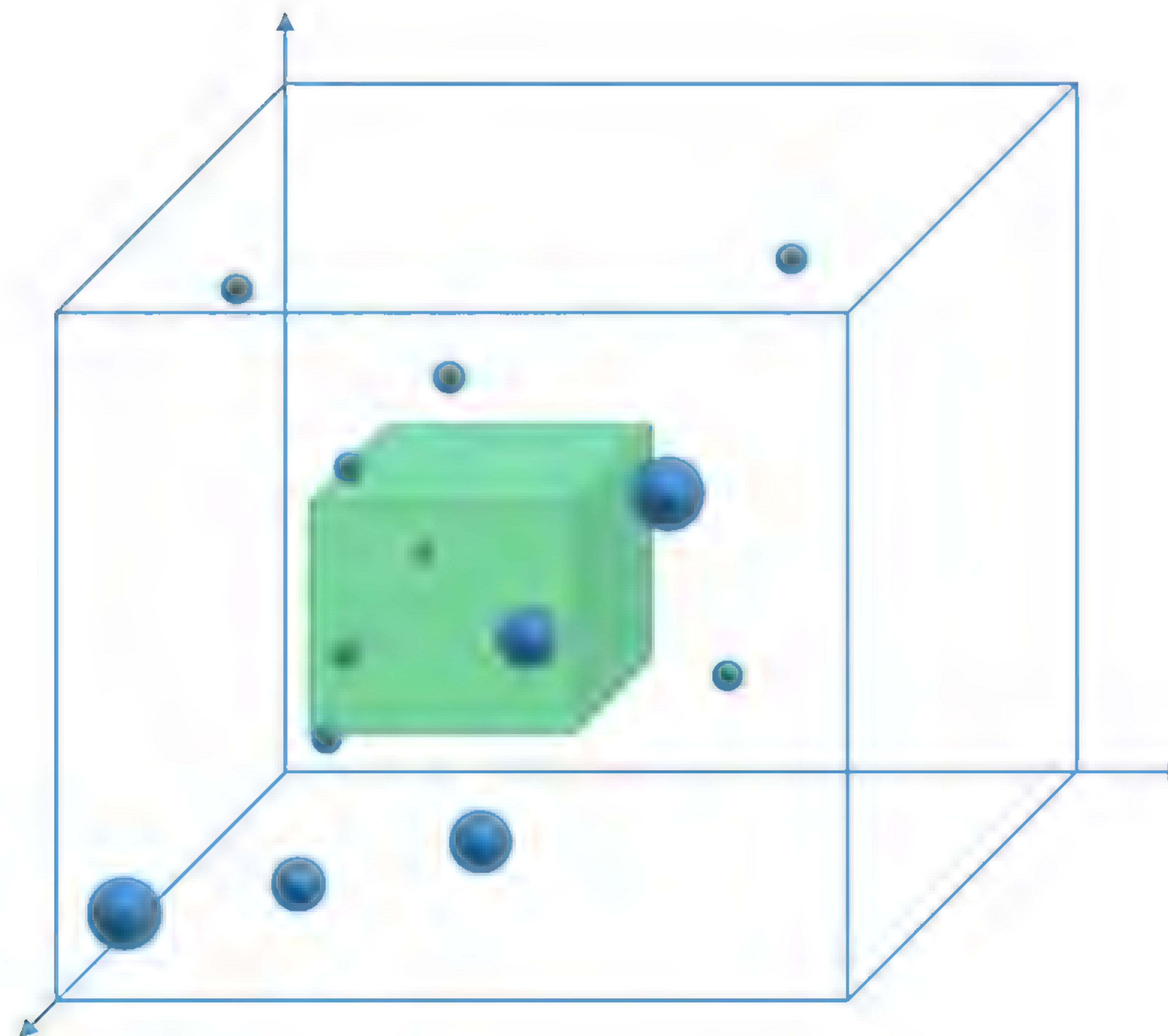
HL HighLoad++
Весна 2021

Тип кузова (4) ▾	Коробка ▾	Дизель ▾	Полный ▾	Р. ▾	К. ▾
0.6 л ▾	2.2 л ▾	Модель от 1980 до ▾	Ремонт ▾		
2011 ▾	2018 ▾	Цена от 0 до ▾	Показать 24 модели		

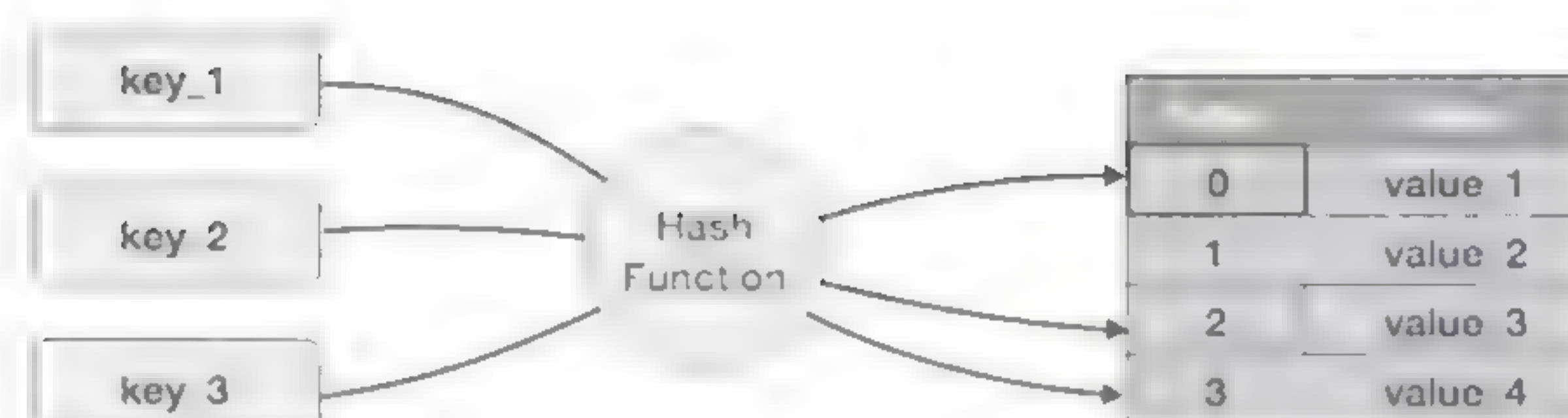
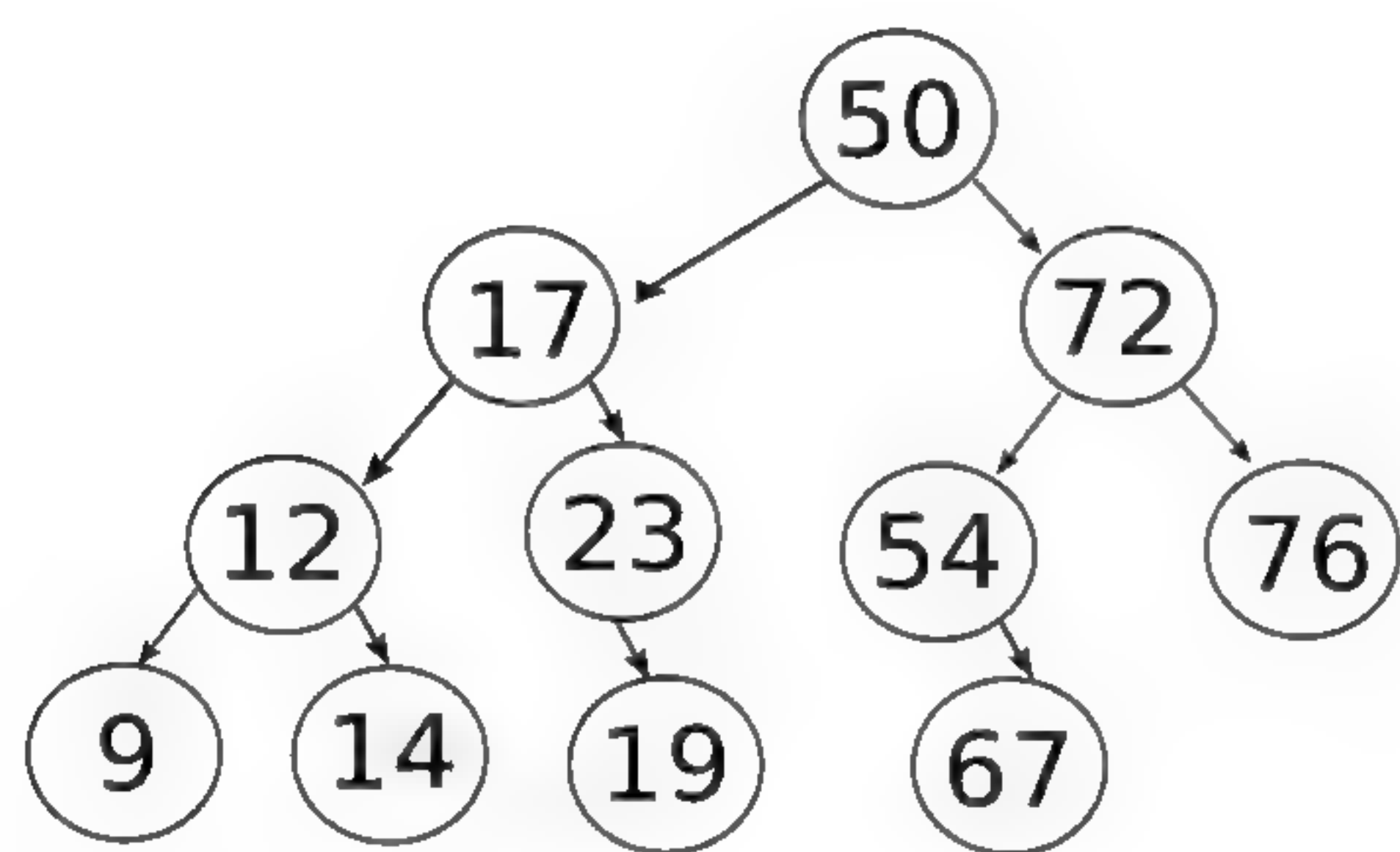
Популярные



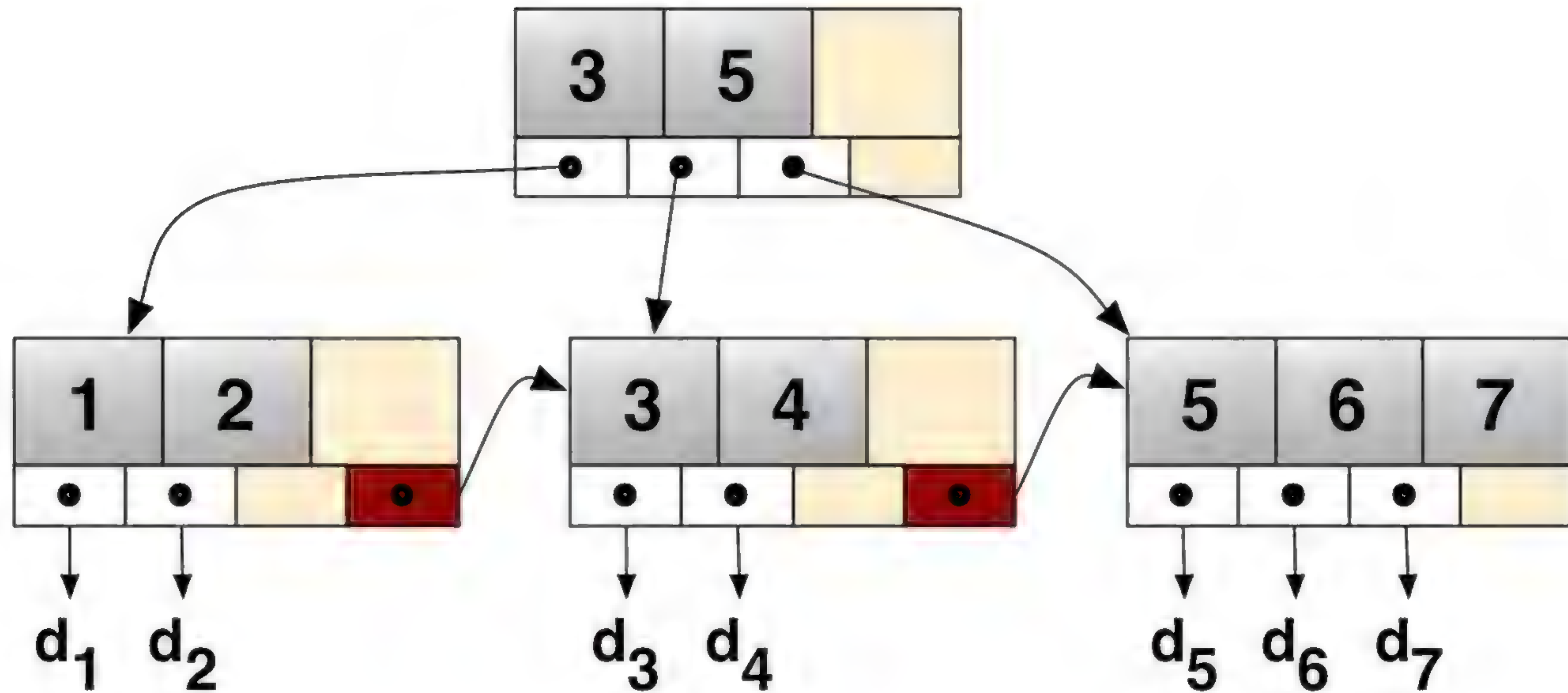
Многокритериальный поиск



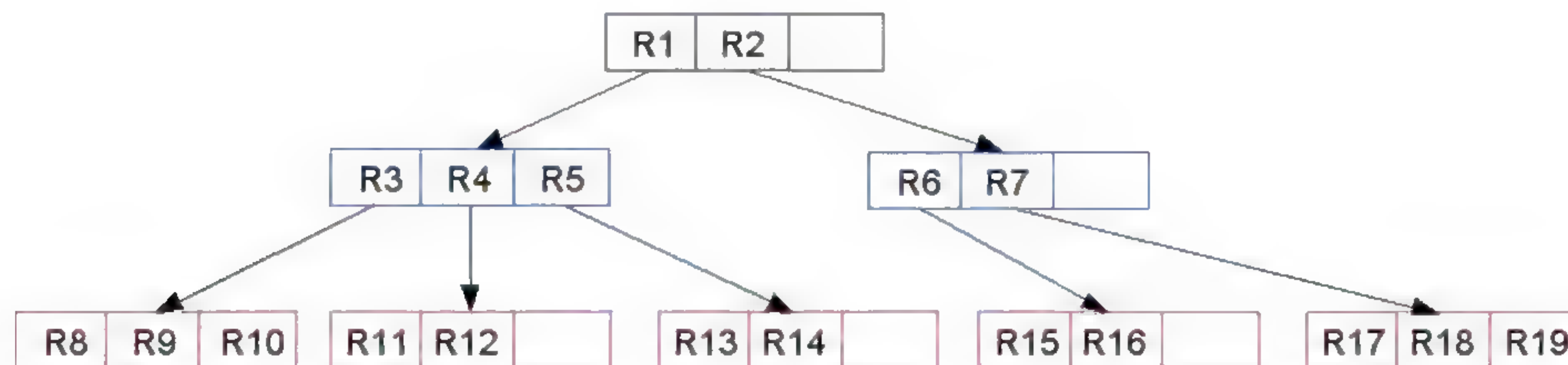
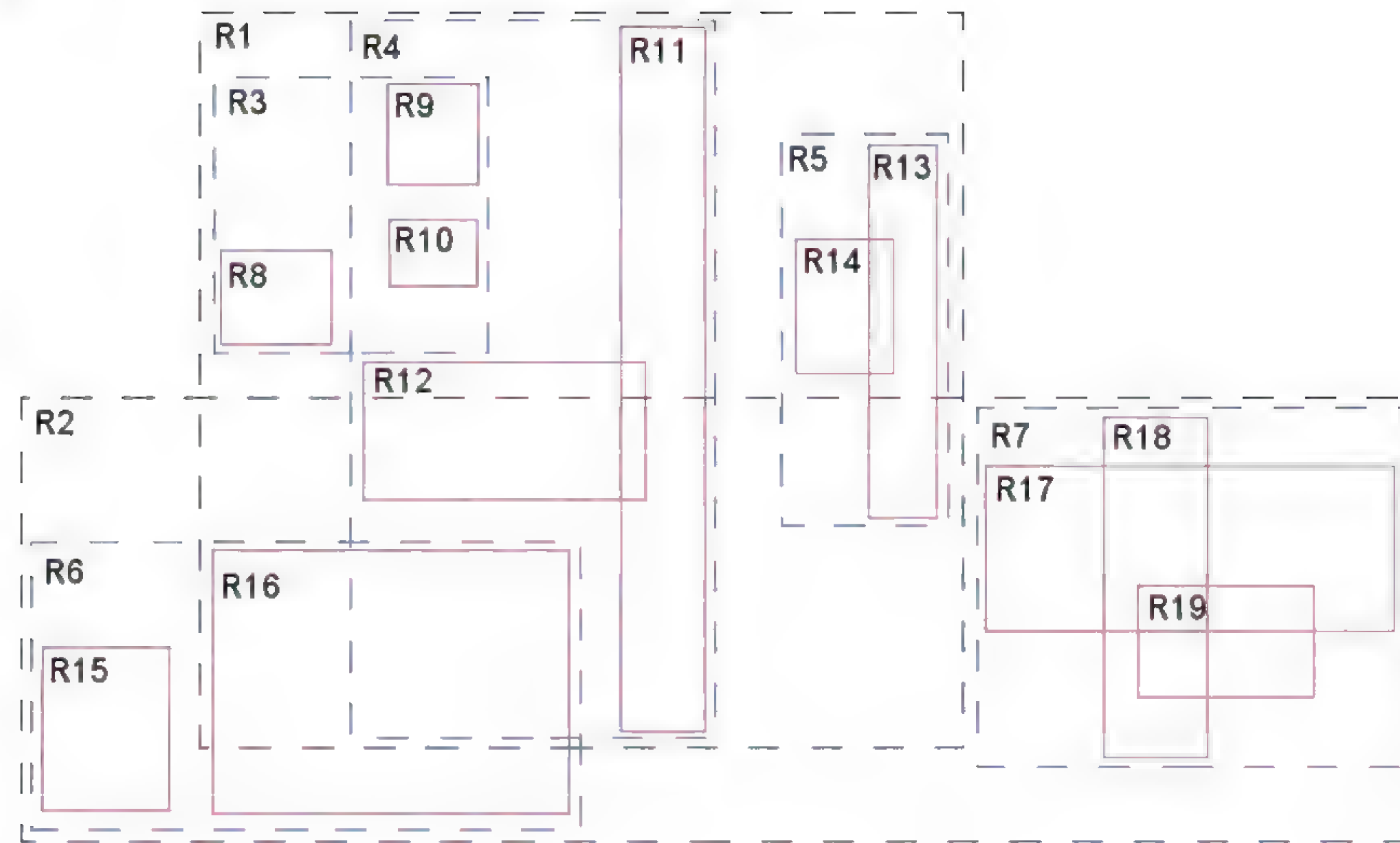
Индексы



B-Tree



R-Tree



Как сейчас?

```
1 space:create_index('primary', {
2     type = 'tree',
3     parts = {{1, 'unsigned'}}
4 })
5 for i = 0, 4 do
6     for j = 0, 4 do
7         space:insert({i * 5 + j, i, j})
8     end
9 end
```

4				
3				
2				
1				
0	1	2	3	4

Наивный подход

```
1 result = {}
2 for _, tuple in space:pairs() do
3     local x = tuple[2]
4     local y = tuple[3]
5     if (1 <= x and x <= 2) and (1 <= y and y <= 2) then
6         table.insert(result, tuple)
7     end
8 end
```


B-Tree

```
1 index = space:create_index('secondary', {
2     type = 'tree',
3     parts = {{2, 'unsigned'}, {3, 'unsigned'}}
4 })
5
6 result = {}
7 for __, tuple in index:pairs({1, 1}, 'GE') do
8     local x = tuple[2]
9     local y = tuple[3]
10
11     if x >= 2 and y > 2 then
12         break
13     end
14
15     if 1 <= y and y <= 2 then
16         table.insert(result, tuple)
17     end
18 end
```


B-Tree

```
1 index = space:create_index('secondary', {
2     type = 'tree',
3     parts = {{2, 'unsigned'}, {3, 'unsigned'}}
4 })
5
6 result = {}
7 for _, tuple in index:pairs({1, 1}, 'GE') do
8     local x = tuple[2]
9     local y = tuple[3]
10
11     if x >= 2 and y > 2 then
12         break
13     end
14
15     if 1 <= y and y <= 2 then
16         table.insert(result, tuple)
17     end
18 end
```


R-Tree

```
1 space:create_index('primary', {
2     type = 'tree',
3     parts = {{1, 'unsigned'}}
4 })
5
6 for i = 0, 4 do
7     for j = 0, 4 do
8         space:insert({i * 5 + j, {i, j}})
9     end
10 end
11
12 index = space:create_index('secondary', {
13     type = 'rtree',
14     parts = {{2, 'array'}},
15     dimension = 2,
16 })
17
18 result = index:select({1, 1, 2, 2}, 'IS')
```


R-Tree

```
1 space:create_index('primary', {
2     type = 'rtree',
3     parts = {{1, 'unsigned'}}
4 })
5
6 for i = 0, 4 do
7     for j = 0, 4 do
8         space:insert({i * 5 + j, {i, j}})
9     end
10 end
11
12 index = space:create_index('secondary', {
13     type = 'rtree',
14     parts = {{2, 'array'}},
15     dimension = 2,
16 })
17
18 result = index:select({1, 1, 2, 2}, 'rtree')
```

R-Tree

```
1 space:create_index('primary', {
2     type = 'rtree',
3     parts = {{1, 'unsigned'}}
4 })
5
6 for i = 0, 4 do
7     for j = 0, 4 do
8         space:insert({i * 5 + j, {i, j}})
9     end
10 end
11
12 index = space:create_index('secondary', {
13     type = 'rtree',
14     parts = {{2, 'array'}},
15     dimension = 2,
16 })
17
18 result = index:select({1, 1, 2, 2}, 'LE')
```


R-Tree

```
1 space:create_index('primary', {
2     type = 'rtree',
3     parts = {{1, 'unsigned'}}
4 })
5
6 for i = 0, 4 do
7     for j = 0, 4 do
8         space:insert({i * 5 + j, {i, j}})
9     end
10 end
11
12 index = space:create_index('secondary', {
13     type = 'rtree',
14     parts = {{2, 'array'}},
15     dimension = 2,
16 })
17
18 result = index:select({1, 1, 2, 2}, 'LE')
```

R-Tree

```
1 space:create_index('primary', {
2     type = 'tree',
3     parts = {{1, 'unsigned'}}
4 })
5
6 for i = 0, 4 do
7     for j = 0, 4 do
8         space:insert({i * 5 + j, {i, j}})
9     end
10 end
11
12 index = space:create_index('secondary', {
13     type = 'rtree',
14     parts = {{2, 'array'}},
15     dimension = 2,
16 })
17
18 result = index:select({1, 1, 2, 2}, 'LE')
```


Кривая Z-порядка

	x: 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
y: 0 000	000000 000001 000100 000101 010000 010001 010100 010101							
1 001	000010 000011 000110 000111 010010 010011 010110 010111							
2 010	001000 001001 001100 001101 011000 011001 011100 011101							
3 011	001010 001011 001110 001111 011010 011011 011110 011111							
4 100	100000 100001 100100 100101 110000 110001 110100 110101							
5 101	100010 100011 100110 100111 110010 110011 110110 110111							
6 110	101000 101001 101100 101101 111000 111001 111100 111101							
7 111	101010 101011 101110 101111 111010 111011 111110 111111							

y = 221

1	1	0	1	1	1	0	1
---	---	---	---	---	---	---	---

x = 73

0	1	0	0	1	0	0	1
---	---	---	---	---	---	---	---

1	0	1	1	0	0	1	0	1	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

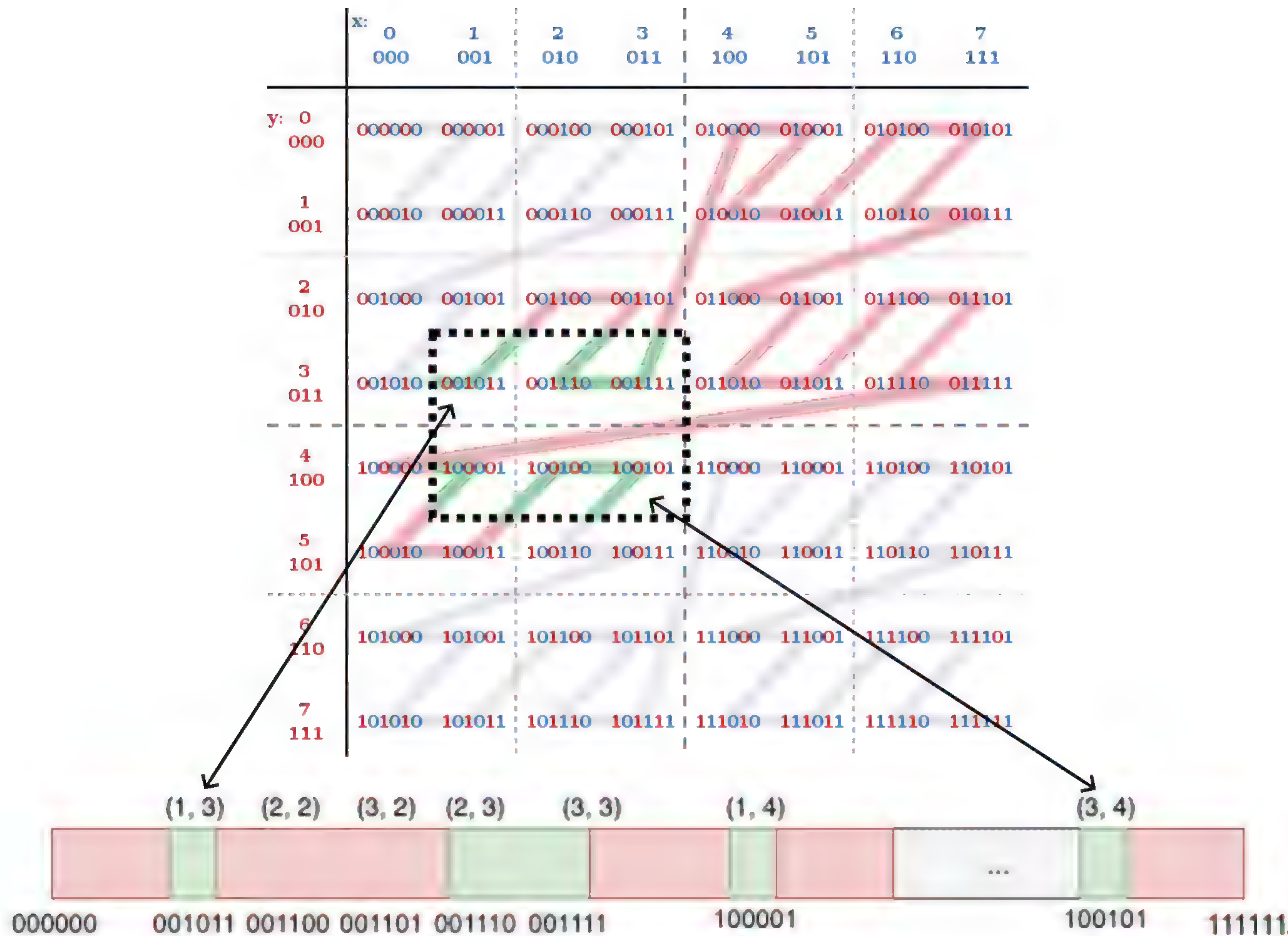
z-address = 45,795

Поиск

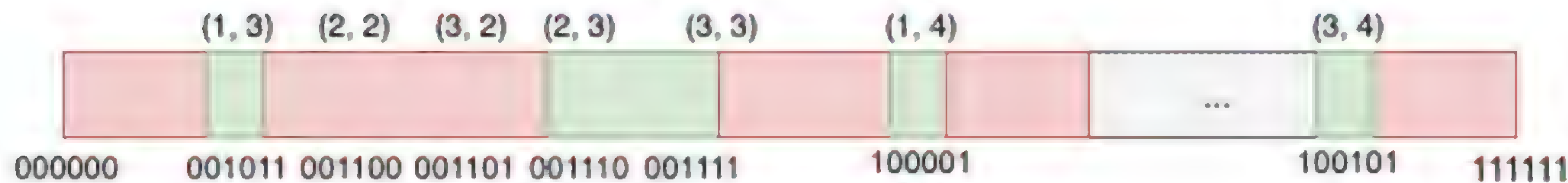
	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111

	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111



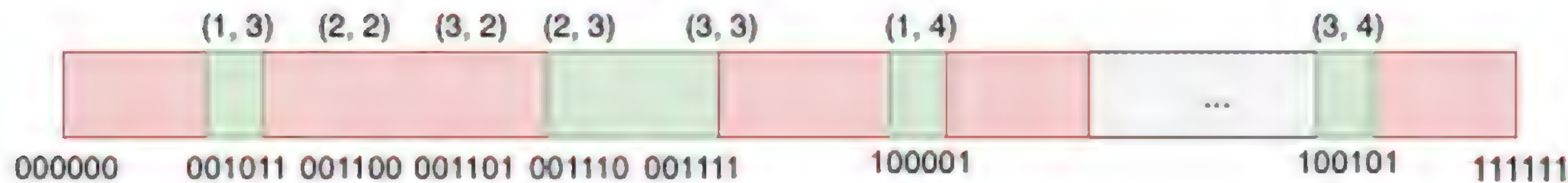


	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111



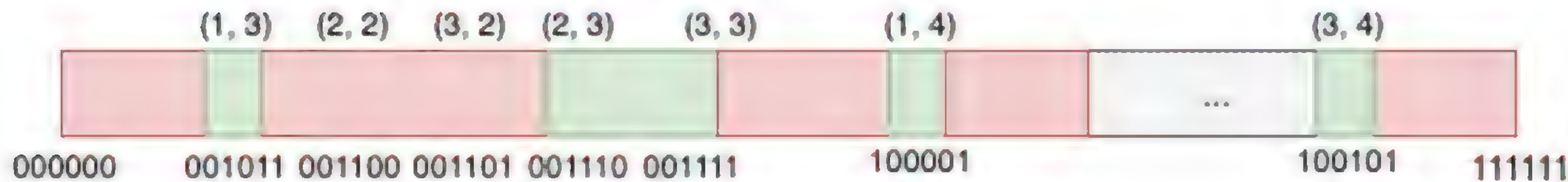
lower bound

	x: 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
y: 0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001000	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111



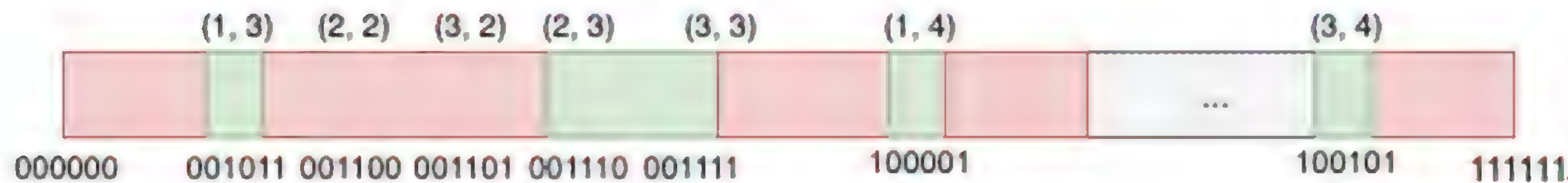
выход за границу

	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111



возвращаемся в область поиска

	x: 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
y: 0 000	000000 000001 000100 000101 010000 010001 010100 010101							
1 001	000010 000011 000110 000111 010010 010011 010110 010111							
2 010	001000 001001 001100 001101 011000 011001 011100 011101							
3 011	001010 001011 001110 001111 011010 011011 011110 011111							
4 100	100000 100001 100100 100101 110000 110001 110100 110101							
5 101	100010 100011 100110 100111 110010 110011 110110 110111							
6 110	101000 101001 101100 101101 111000 111001 111100 111101							
7 111	101010 101011 101110 101111 111010 111011 111110 111111							



upper bound

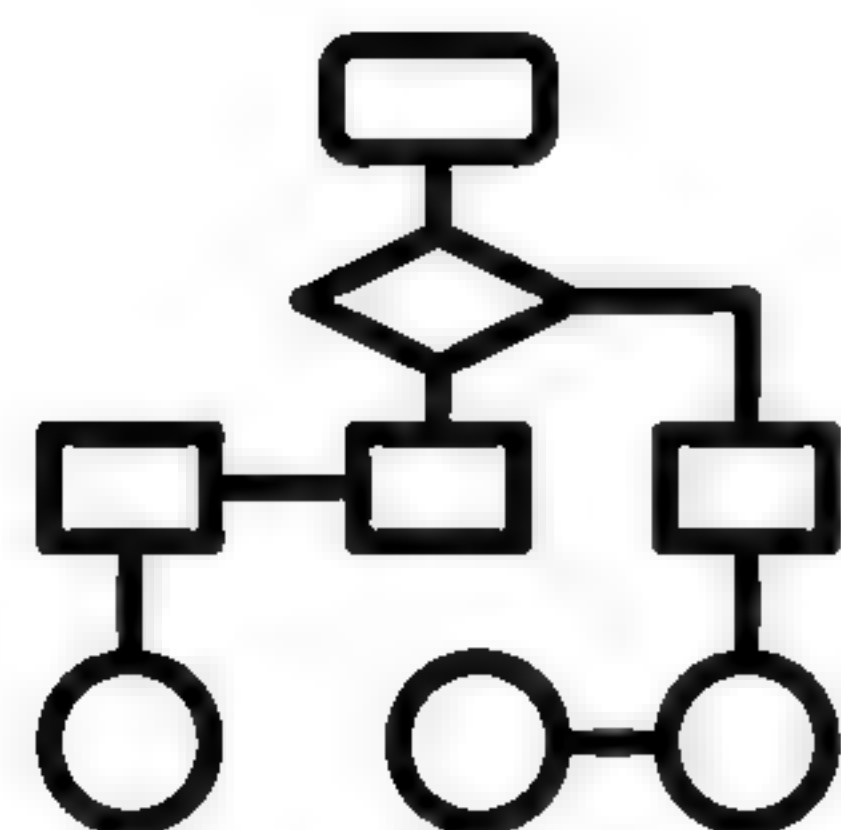
Встраивание в Tarantool

- Вычисление z-адреса
- Сохранение в B+*-Tree
- Реализация алгоритмов поиска

Что доступно разработчику?



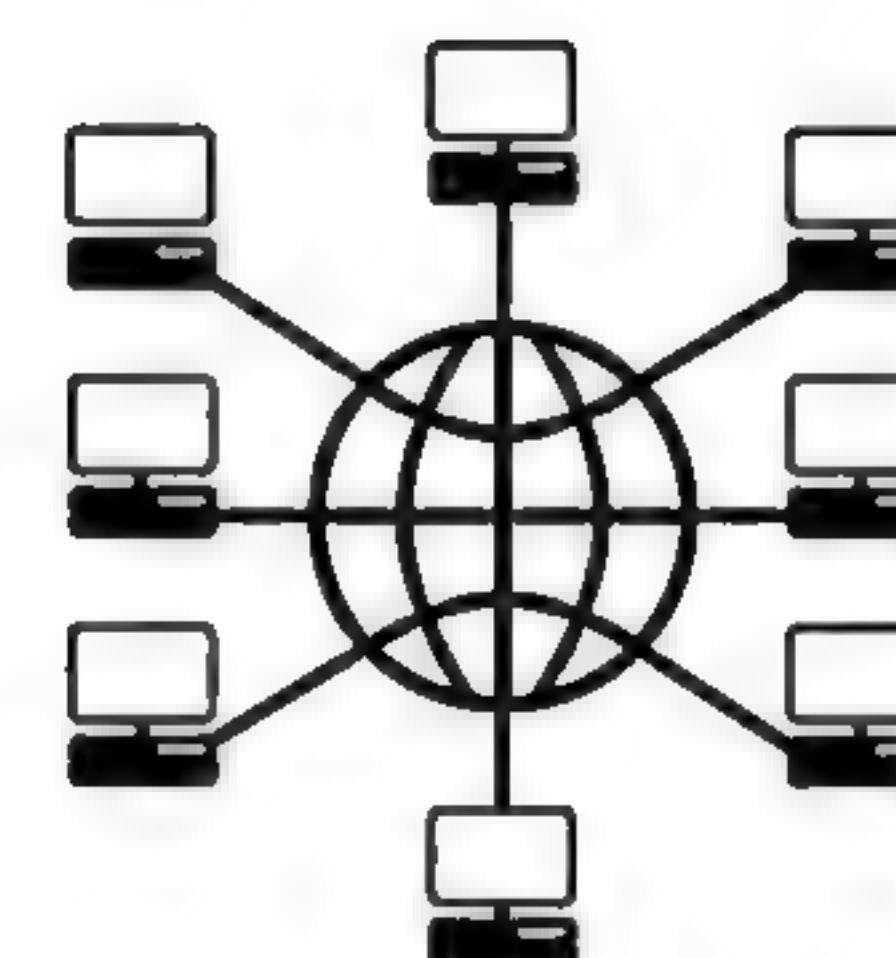
Small



Salad



C/Lua API

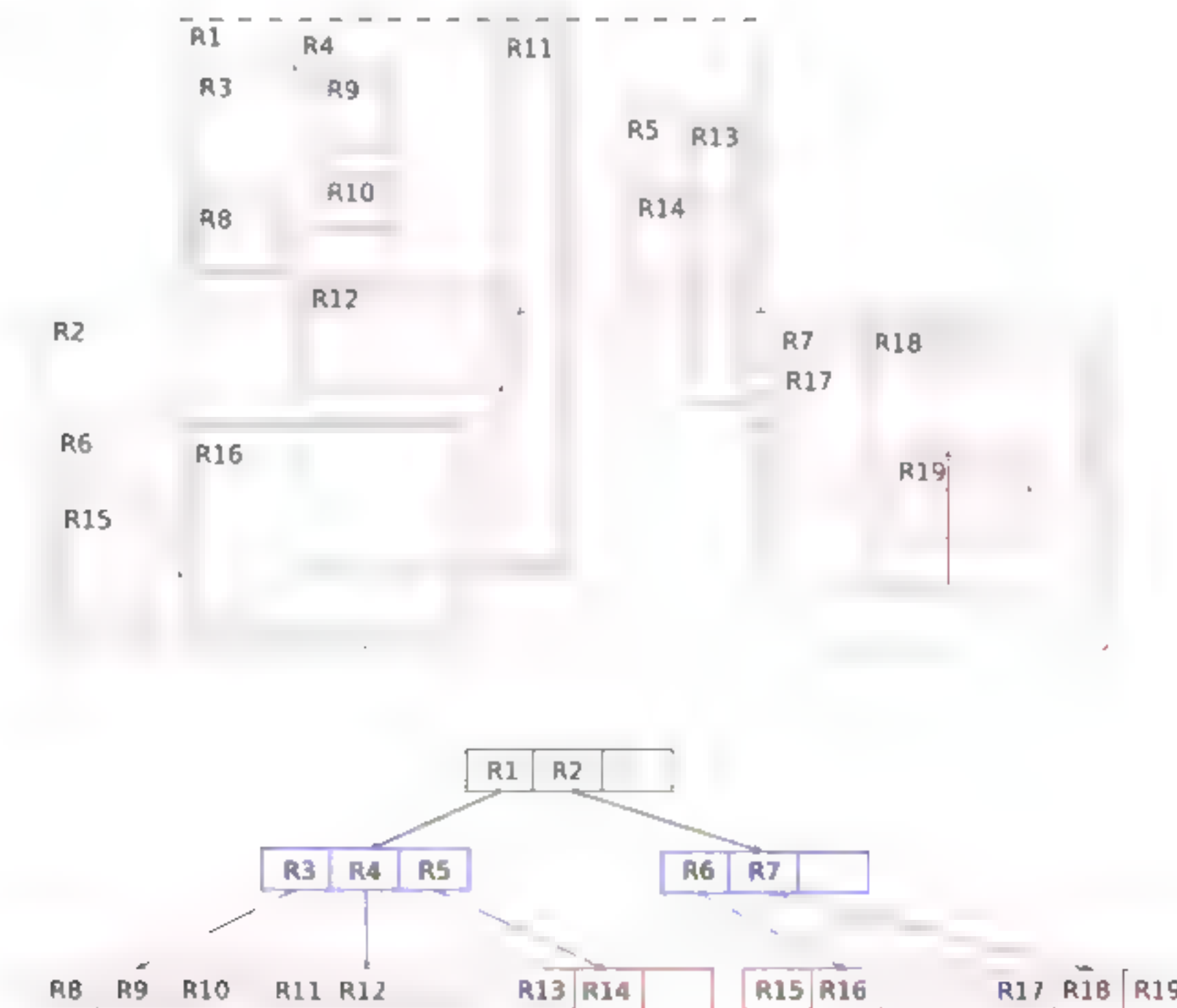
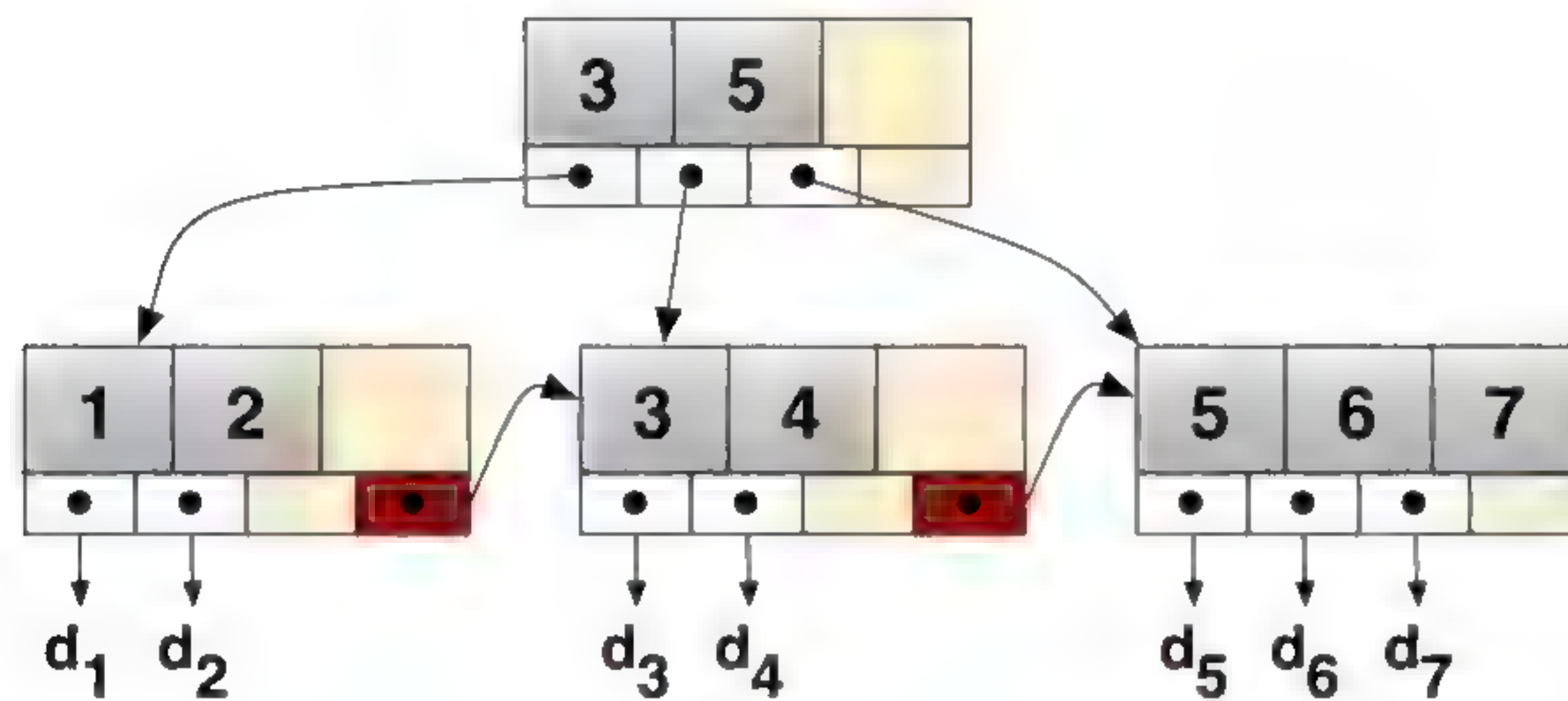
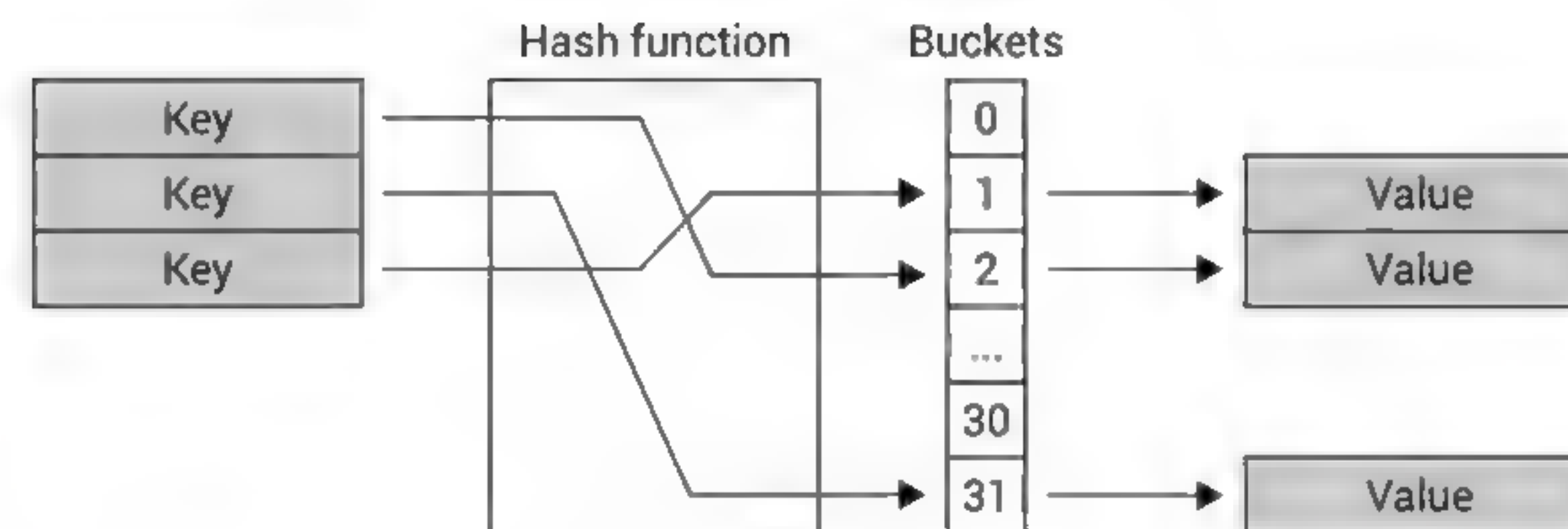


Network

Specialized Memory ALLocators



Some Algorithms And Data structures



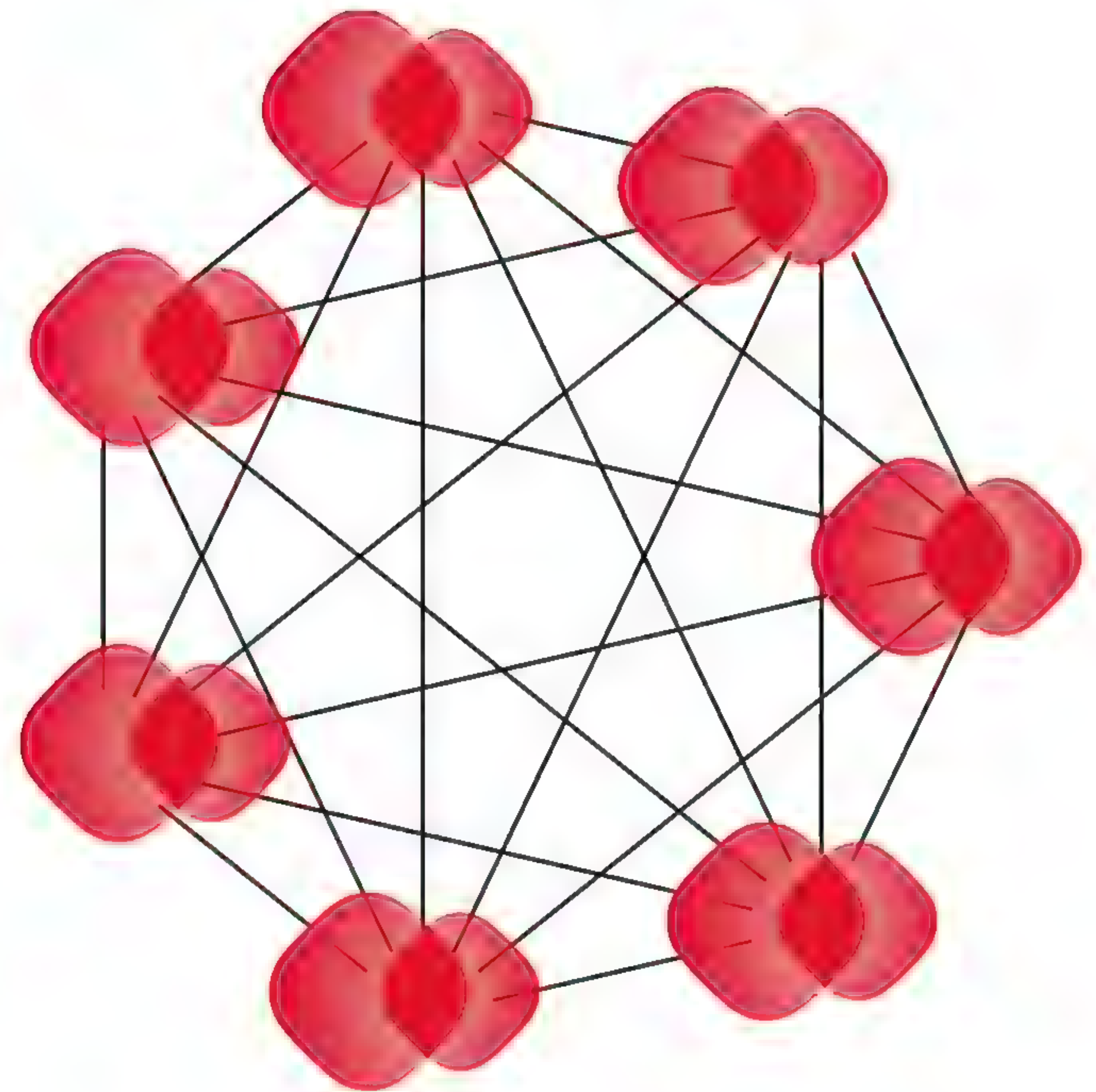
C API/Lua API

- Lua C API
- LuaJIT FFI
- Tarantool C API



Network

- IProto
 - Call
 - Replication
 - DML
 - RAFT
- SWIM
- ...



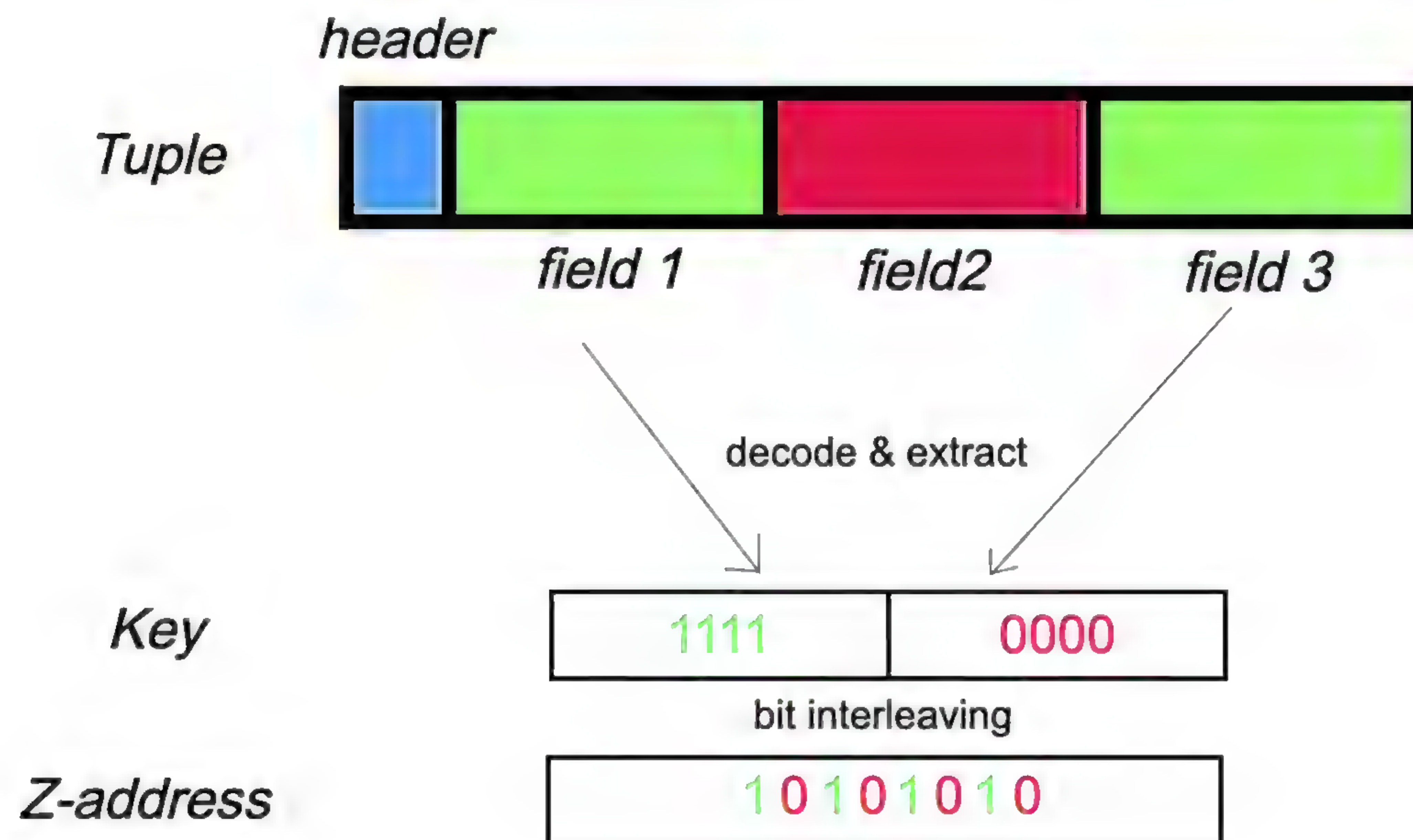
Что понадобилось?

- Набор алгоритмов для работы с Z-адресами
- B+*-Tree
- Bit array

Работа с Z-адресами



Вычисление



Хранение в BPS-Tree

```
struct memtx_tree_data {  
    /* Tuple that this node is represents. */  
    struct tuple *tuple;  
    /** Comparison hint, see key_hint(). */  
    hint_t hint;  
};
```



```
struct memtx_zcurve_data {  
    /* Z-address. Read here: https://en.wikipedia.org/wiki/Z-order\_curve */  
    z_address *z_address;  
    /** Tuple that this node is represents. */  
    struct tuple *tuple;  
};
```

Работа с типами

Type	Dec	Bin	Normalized
unsigned	6	00000110	00000110
integer	6	00000110	10000110
	-6	10000110	00000110
string	abc	0x61,0x62,0x63	0x61,0x62
	a	0x61	0x61,0x00

Next-jump-in

	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0 000		000000	000001	000100	000101	010000	010001	010100	010101
1 001		000010	000011	000110	000111	010010	010011	010110	010111
2 010		001000	001001	001100	001101	011000	011001	011100	011101
3 011		001010	001011	001110	001111	011010	011011	011110	011111
4 100		100000	100001	100100	100101	110000	110001	110100	110101
5 101		100010	100011	100110	100111	110010	110011	110110	110111
6 110		101000	101001	101100	101101	111000	111001	111100	111101
7 111		101010	101011	101110	101111	111010	111011	111110	111111

lower bound - 11
upper bound - 50

	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111

lower bound - 11
upper bound - 50

	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111

lower bound - 11
upper bound - 50
current - 40


	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111

lower bound - 11
upper bound - 50
current - 40
NIP - 48

Индекс в Tarantool


- replace
- create_iterator
- get
- count
- size
- bsize
- min
- max
- update_def
- depends_on_pk
- ...

Replace



```
1 int
2 memtx_zcurve_index_replace(struct index *base, struct tuple *old_tuple,
3                             struct tuple *new_tuple, enum dup_replace_mode mode,
4                             struct tuple **result)
```


Create iterator



```
1 struct iterator *  
2 memtx_zcurve_index_create_iterator(struct index *base,  
3     enum iterator_type type, const char *key, uint32_t part_count)
```

Lua API

```
1 space = box.schema.space.create('my_space')
2 space:create_index('primary',
3     {type = 'tree', parts = {{1, 'unsigned'}}})
4 space:create_index('secondary',
5     {type = 'btree', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
6 for i = 0, 5 do
7     for j = 0, 5 do
8         space:insert({i * 6 + j, i, j})
9     end
10 end
```

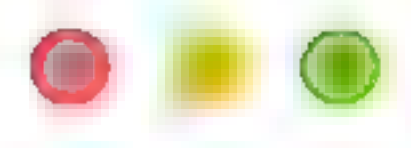

Lua API

```
1 space = box.schema.space.create( my_space )
2 space:create_index('primary',
3     {type = 'tree', parts = {{1, 'unsigned'}}})
4 space:create_index('secondary',
5     {type = 'zcurve', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
6 for i = 0, 5 do
7     for j = 0, 5 do
8         space:insert({i * 6 + j, i, j})
9     end
10 end
```

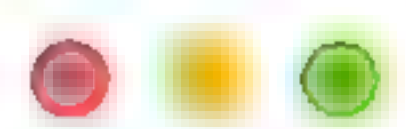
Lua API

```
1 space = box.schema.space.create( 'my_space' )
2 space:create_index('primary',
3     {type = 'tree', parts = {{1, 'unsigned'}}})
4 space:create_index('secondary',
5     {type = 'zcurve', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
6 for i = 0, 5 do
7     for j = 0, 5 do
8         space:insert({i * 6 + j, i, j})
9     end
10 end
```

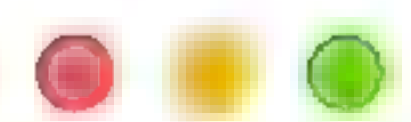

Lua API



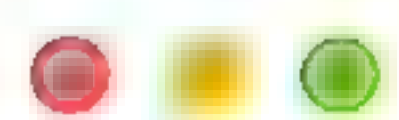
```
1 space = box.schema.space.create('my_space')
2 space:create_index('primary',
3     {type = 'tree', parts = {{1, 'unsigned'}}})
4 space:create_index('secondary',
5     {type = 'zcurve', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
6 for i = 0, 5 do
7     for j = 0, 5 do
8         space:insert({i * 6 + j, i, j})
9     end
10 end
```



```
1  -- (2 <= x <= 3) and (3 <= y <= 5)
2  tarantool> secondary:select({2, 3, 3, 5})
3  ---
4  - - [15, 2, 3]
5      - [21, 3, 3]
6      - [16, 2, 4]
7      - [22, 3, 4]
8      - [17, 2, 5]
9      - [23, 3, 5]
10 ...
```

```
1  -- (x == 2) and (y == 3)
2  tarantool> secondary:select({2, 3})
3  ---
4  - - [15, 2, 3]
5  ...
```



```
1  -- (2 <= x <= 3)
2  tarantool> secondary:select({2, 3, box.NULL, box.NULL})
3  ---
4  - - [12, 2, 0]
5      - [18, 3, 0]
6      - [13, 2, 1]
7      - [19, 3, 1]
8      - [14, 2, 2]
9      - [20, 3, 2]
10     - [15, 2, 3]
11     - [21, 3, 3]
12     - [16, 2, 4]
13     - [22, 3, 4]
14     - [17, 2, 5]
15     - [23, 3, 5]
16  ...
```




```
1  -- (x >= 2) and (y >= 3)
2  tarantool> secondary:select({2, box.NULL, 3, box.NULL})
3  ---
4  - - [15, 2, 3]
5      - [21, 3, 3]
6      - [27, 4, 3]
7      - [33, 5, 3]
8      - [16, 2, 4]
9      - [22, 3, 4]
10     - [17, 2, 5]
11     - [23, 3, 5]
12     - [28, 4, 4]
13     - [34, 5, 4]
14     - [29, 4, 5]
15     - [35, 5, 5]
16  ...
```

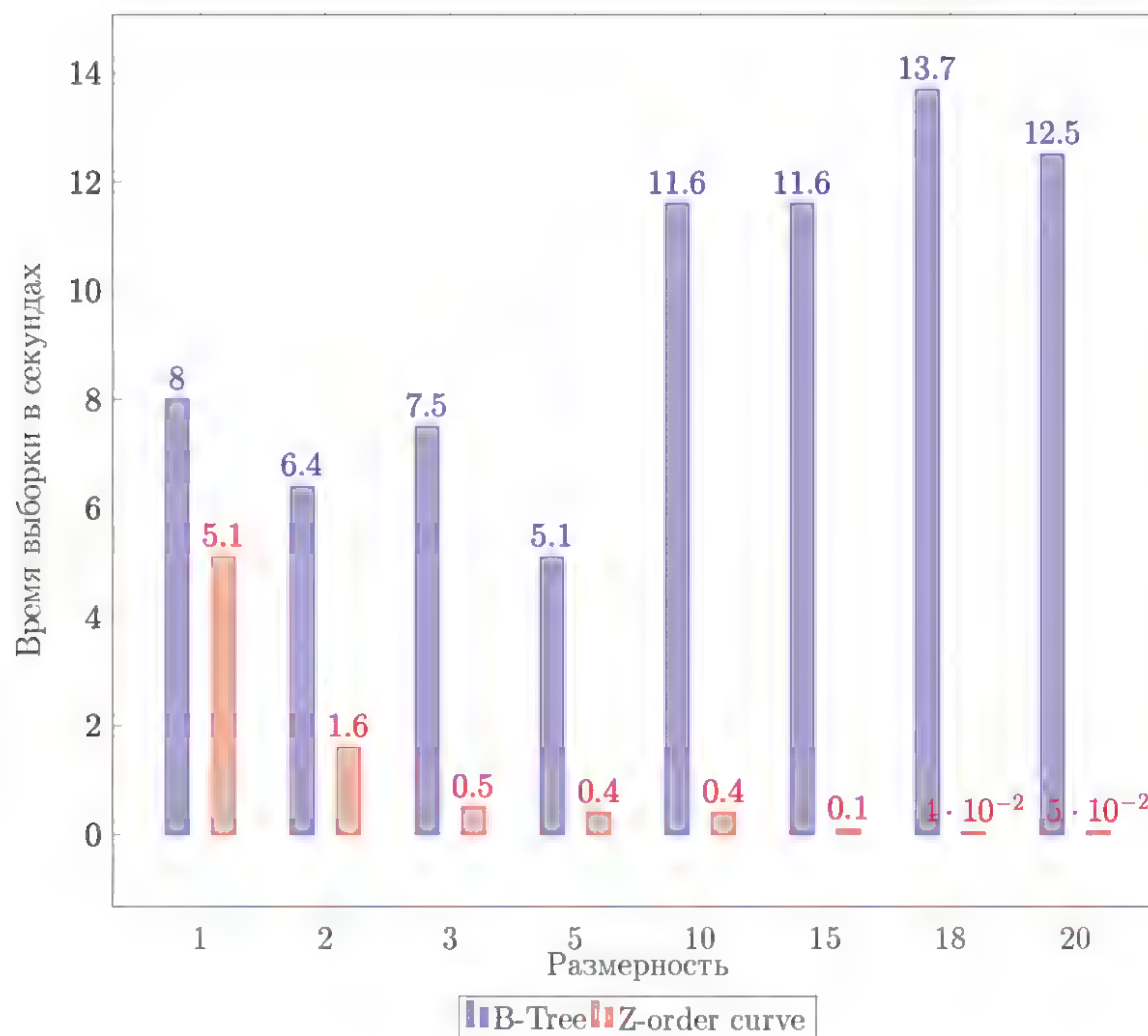


Z-order curve vs B-Tree

- Поиск в прямоугольной области
- Префиксный поиск

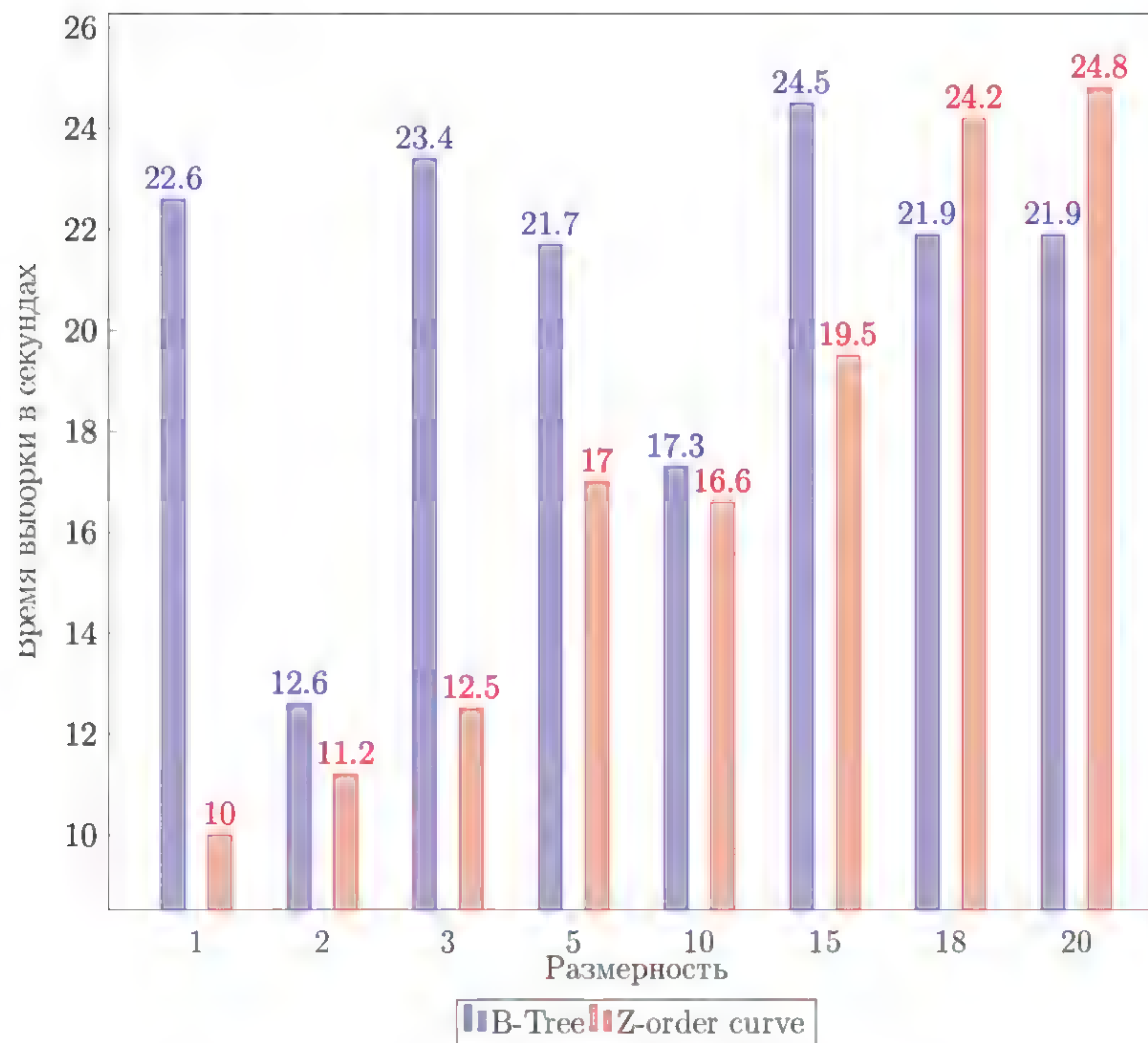
Поиск в прямоугольной области

10^6 точек, распределенных равномерно



Поиск в прямоугольной области

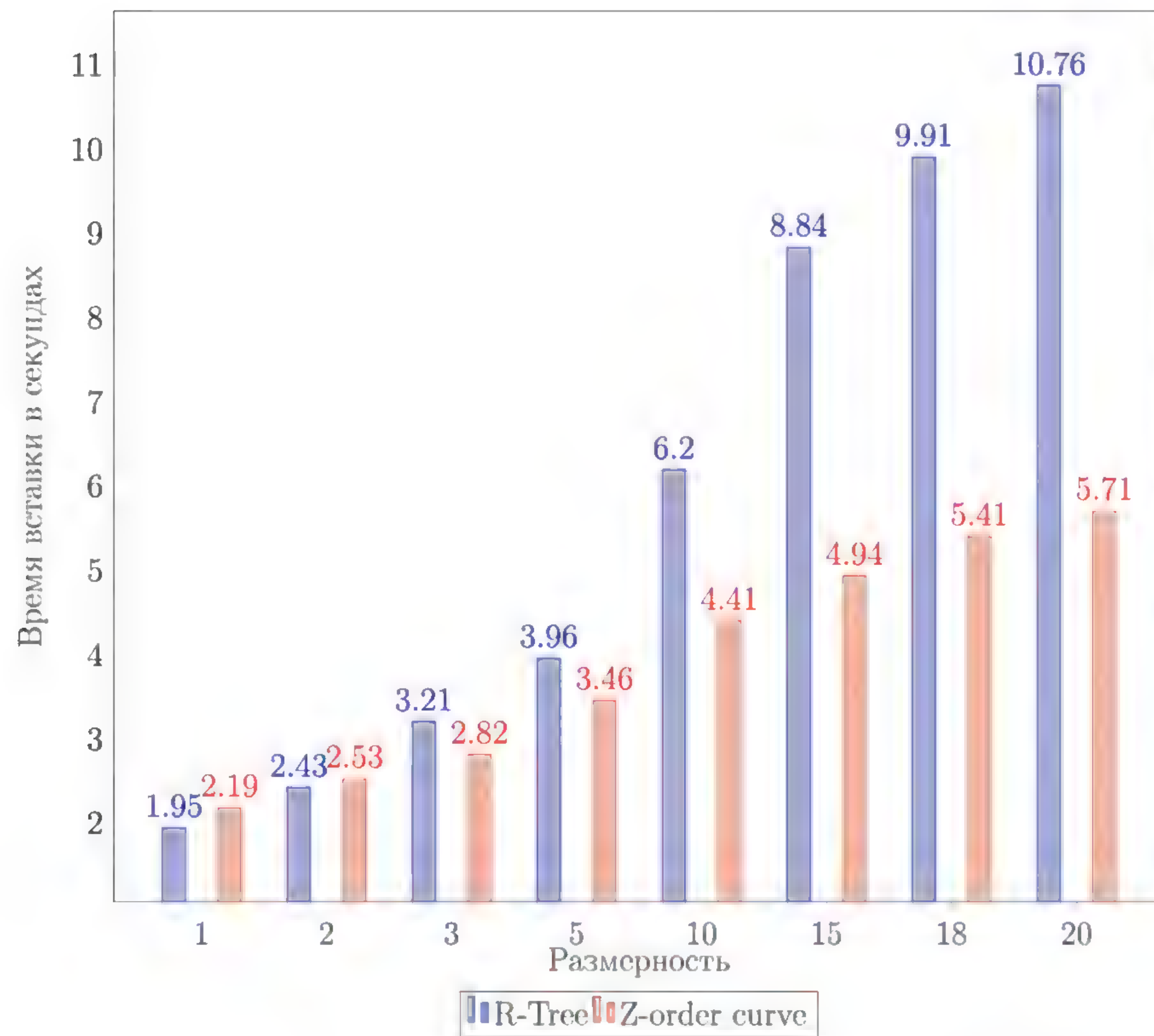
10^6 точек, смещенных к 0



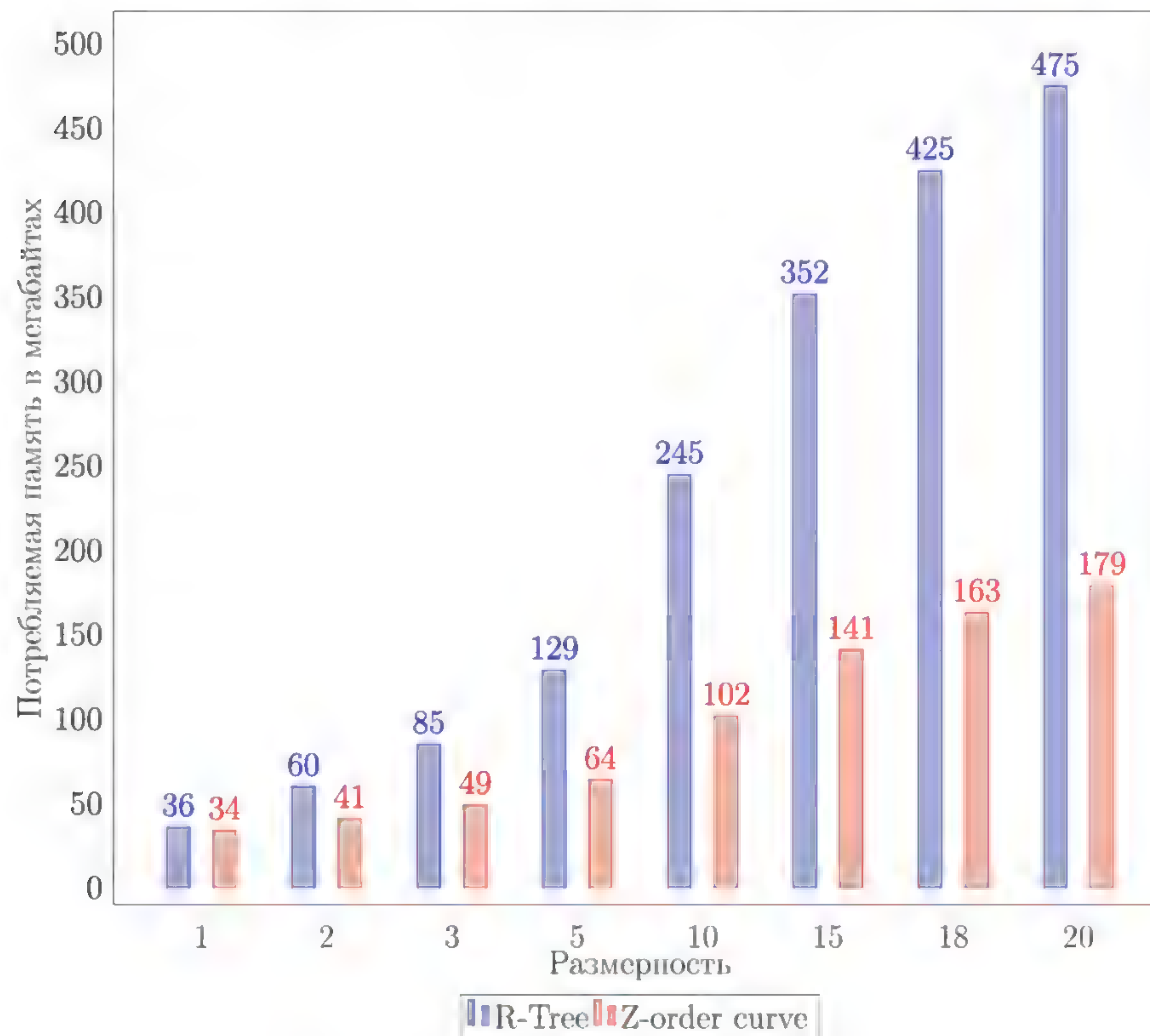
Z-order curve vs R-Tree

- Запись
- Потребляемая память
- Поиск в прямоугольной области

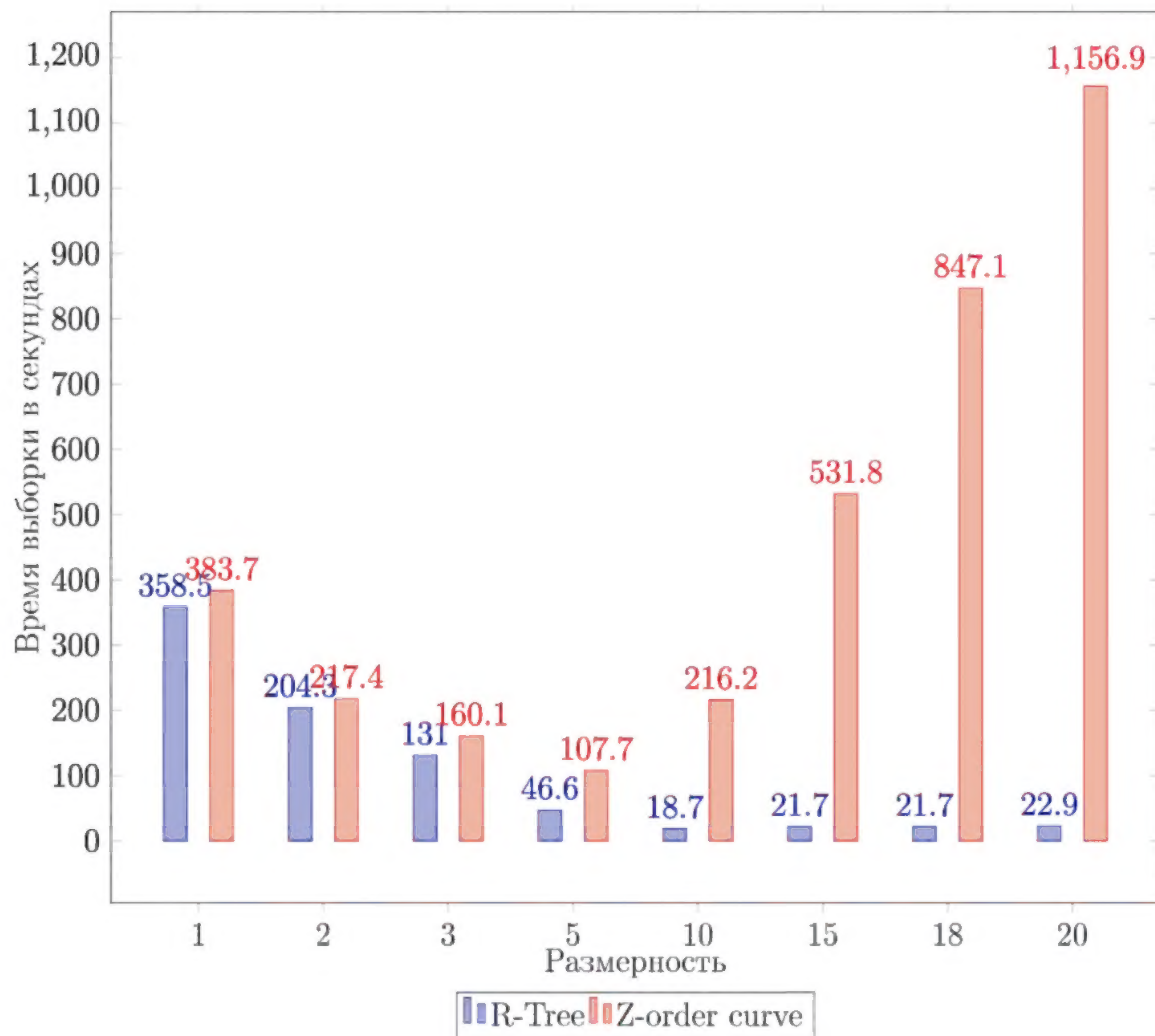
Запись



Потребление памяти



Поиск в прямоугольной области



Почему за год всё сломалось?



alexlyapunov 8 февраля 2021 в 16:45

Менеджер транзакций для базы данных в оперативной памяти

Блог компании Mail.ru Group, Высокая производительность, Алгоритмы, Хранение данных, Tarantool



highLoad++
весна 2021



Вывод

- Отрицательный результат — тоже результат :)
- Tarantool — отличная платформа для изучения БД
- Не бойтесь экспериментировать

Вопросы

Олег Бабин

tg: @olegrok

e-mail: o.babin@corp.mail.ru

Fork:

<https://github.com/olegrok/tarantool/tree/z-order-curve-index>